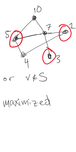
Max Weight Independent Set Problem (MWIS)

Input: Graph: $G=(V,E)$ (vertices, edges)
Weights: $w: V \to \mathbb{Z}^+$ (positive integers)



Output: $S \subseteq V$ s.t.

Independent Set ⇒ • If $\{u,v\} \in E$, $u \notin S$ or $v \notin S$

Max Weight ⇒ • $W(S) = \sum_{v \in S} w(v)$ is maximized

objective function

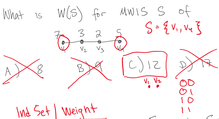
Applications

- Cell Tower Transmission
- Choose Franchise Location
- Party Invite
- Scheduling

* General graph → very hard problem

* Only look at the graph

What is $W(S)$ for MWIS S of $S = \{v_1, v_4\}$



Ind Set	Weight
$\emptyset$	0
$\{v_1\}$	7
$\{v_1, v_4\}$	9
~~$\{v_2, v_3\}$~~	~~8~~
⋮	⋮

n-vertex graph

For each $S \subseteq V$ ⇐ $O(2^n)$
  If Ind. Set: ⇐ $O(n^2)$
    Check Weight ⇐ $O(n)$
    Store if largest seen ⇐ $O(1)$

$O(n^2 \cdot 2^n)$ $\quad O(c \cdot 2^n)$

Settling in? Change in tutoring/office hours? Talk today! WICS++ Wed! Hacking Thurs!

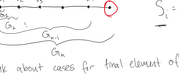
Goals:
- Design a dynamic programming algorithm for MWIS

Qs: Big idea with greedy proof. Why "greedy"? More than two vertices?

Divide + Conquer... better but not best

Designing a Dynamic Programming Alg

0. Find series of increasingly smaller similar subproblems
   Recurrence Object: Optimal output of each subproblem.



1. Think about cases for final element of recurrence obj.

   S_n
   i) $v_n \notin S_n$
   ii) $v_n \in S_n$

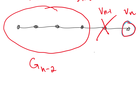
2. For each case, create a recurrence

   Options:
   S_{n-1}
   S_{n-2}
   $S_{n-1} \cup \{v_n\}$
   $S_{n-2} \cup \{v_{n-1}\}$
   $S_{n-2} \cup \{v_n\}$

   i) If $v_n \notin S_n$, $S_n = S_{n-1}$

   ii) If $v_n \in S_n$, $S_n = S_{n-2} \cup \{v_n\}$

(i)



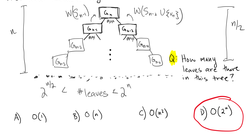
(ii)

Conclusion

$S_n = \begin{cases} S_{n-1} \\ \text{or} \\ S_{n-2} \cup \{v_n\} \end{cases}$

ONLY Possibilities
Evaluate both + Take better option
↓
Larger weight

First idea: Recursive alg



Q: How many leaves and more in this tree?

A) $O(1)$ B) $O(n)$ c) $O(n^2)$ D) $O(2^n)$

* Actually solving same problems over and over!

Q: How many distinct subproblems are there?

A) $O(1)$ B) $O(n)$ c) $O(n^2)$ D) $O(2^n)$

$G_n, G_{n-1}, G_{n-2} \cdots G_1$

Idea: Instead of solving recursively, store solutions in an array, look up

Trick 1: Build up instead of look back

Trick 2: Store objective function value instead of set/strategy object (faster)

$S_n = \begin{cases} S_{n-1} \\ S_{n-2} \cup \{v_n\} \end{cases} \iff W(S_n) = \begin{cases} W(S_{n-1}) \\ W(S_{n-2}) + W(v_n) \end{cases}$

MWIS on Line

// Determine $W(S)$'s:
1.
2.
3.

// Determine MWIS S_n
4. $S_n \leftarrow \emptyset$
5. $i \leftarrow n$
6. while $i \geq 1$:
   if $A[i] = A[i-1]$:
   ?
   else:
   ?

ex: 5 7 4 4 1
v_1 v_2 v_3 v_4 v_5

S = { }

1. Fill in A from example, using code
3. Fill in S from example, using code.

2. Fill in code

4. Runtime

Runtime:
Proof:
Ethics:

Goals:
- Design a dynamic programming algorithm for MWIS

Reminders:
- Reflection, Fill out form for tutoring office hour scheduling feedback